Серия «Математика»
2017. Т. 21. С. 89—107

Онлайн-доступ к журналу:
http://mathizv.isu.ru

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 004.82, 510.62
MSC 68T27
DOI https://doi.org/10.26516/1997-7670.2017.21.89

Документное моделирование

А. А. Малых
Иркутский государственный университет

А. В. Манцивода*
Иркутский государственный университет,
Институт математики им. С.Л. Соболева

Аннотация. Рассматривается концепция локально-простых моделей. Локально-простые модели — это сколь угодно сложные модели, выстроенные из относительно простых компонент. Очень много практически значимых предметных областей описываются как локально-простые модели, например, бизнес-модели предприятий и компаний. До сих пор исследования в области автоматизации человеческих рассуждений в основном концентрировались вокруг наиболее интеллектуально насыщенных занятий, в частности, автоматизации доказательства математических теорем. Но, например, модель ритейлера формируется из «рабочих мест», на каждом из которых решаются намного более простые и относительно легко автоматизируемые интеллектуальные задачи. При этом сама модель ритейлера как целостная система исключительно сложна. В данной работе предлагается вариант математического определения концептуального понятия локально-простой модели. Это определение ориентировано на работу в самых разных предметных областях. Поэтому нам важно учесть и перцептивную, психологическую составляющую. Логика элитарна, и если мы хотим, чтобы максимально широкий круг людей работал с нашими моделями, необходимо спрятать эту элитарность за метафорой, привычной «обычным» людям. В качестве такой метафоры мы используем концепт документа, поэтому наши локально-простые модели называются документными моделями. Документные модели строятся в парадигме семантического программирования. Это позволяет достигнуть еще одного важного качества — документные модели являются исполняемыми. Исполняемые модели — это такие модели, которые могут действовать как практические информационные системы в описываемой предметной области. Таким образом, если наша модель исполняемая, то этап программирования становится ненужным. Использование модели напрямую, вместо программного кода, дает очень важные преимущества, например резкое снижение стоимости разработки информационной системы. Кроме того, поскольку модель остается нетронутой,

* Исследование выполнено за счет гранта Российского научного фонда (проект №17-11-01176).

незакодированной в программных модулях, ею непосредственно могут пользоваться средства искусственного интеллекта, в частности, машинного обучения. Это существенно расширяет возможности для автоматизации и роботизации управленческой деятельности.

Ключевые слова: локально простая модель, документная модель, семантическое программирование.

1. Введение

Одним из важных выводов из классических работ по семантическому программированию [6; 7] является то, что описание действий в некоторой предметной области может быть сделано не в императивном стиле (с помощью программ на языках программирования), а в декларативном – через моделирование предметной области в некоторой логической системе. Конечно, такая логическая система должна удовлетворять ряду требований, основным из которых является возможность интерпретации логических описаний как процедур. Авторы концепции семантического программирования сформулировали общий математический подход, объясняющий то, как могли бы выглядеть такие модели. Их мы будем называть *исполняемыми*, поскольку они не только задают декларативное описание предметной области, но и описывают набор действий, которые могут производиться в рамках предметной области. Если модель исполняемая, то она может напрямую работать в качестве информационной системы.

Моделирование предметной области в рамках семантического программирования похоже на процесс разработки проекта некоторой информационной системы. Семантическая модель является результатом такой работы. Исполняемость модели означает, что если мы описали проект некоторой IT-системы, то программировать эту систему уже не надо, поскольку функционал этой системы автоматически строится из семантической модели.

Такая идея выглядит несколько фантастично. Например, если подобное удалось бы сделать в строительстве, то здание появлялось бы на свет сразу после разработки архитектором его проекта (модели здания). Тем не менее, оказалось, что эта идея вполне может быть реализована на практике.

Глобальным следствием этой идеи является то, что

мы можем заменить программирование на моделирование.

Результаты такой замены оказываются весьма впечатляющими:

– Семантическое моделирование существенно менее затратное в разработке и поддержке, чем программирование.

– Моделирование в отличие от программирования доступно намного более широкому кругу специалистов, например, экспертам в моделируемой предметной области.
– В отличие от программ с моделями напрямую могут работать средства искусственного интеллекта и программные роботы.

Первый пункт говорит о том, что семантическое моделирование, там где оно применимо, имеет существенные экономические преимущества по сравнению с классическим программированием. Второй пункт позволяет при работе в некоторой предметной области освободиться от таких посредников, как программисты, поскольку специалисты в данной предметной области могут напрямую заниматься разработками. Третий пункт очень важен в свете IV промышленной революции, свидетелями которой мы сегодня являемся. Он означает, что роли (рабочие места), описываемые моделью, могут быть роботизированы.

Таким образом, если нам удается для некоторой отрасли задействовать средства семантического моделирования, это имеет прорывной эффект, а мы получаем существенные конкурентные преимущества в этой отрасли.

2. Локально-простые модели

Моделирование знаний и автоматизация человеческих рассуждений – направление математической логики и искусственного интеллекта, насчитывающего уже более чем полувековую историю. Наибольший толчок дало изобретение метода резолюций [12]. Он позволил добиться весьма значительных результатов на ниве автоматического доказательства теорем [13]. Использовать возможности моделирования знаний для управления информацией в интернете предложил Тим Бернерс-Ли [5]. Идея состояла в использовании логических средств для более эффективной организации данных в мировой сети и в создании автоматических агентов. Из этой идеи выросло новое направление исследований — семантический веб [14]. В качестве логической основы использовались разнообразные дескриптивные логики [8; 9].

К сожалению, семантический веб не привел к глобальным улучшениям интернета, как на то надеялся Бернерс-Ли. Это направление превратилось в достаточно узкую математическую дисциплину с минимальным влиянием на внешний мир и очень слабым распространением в кругах практиков. Основная проблема с семантическим вебом состояла, по нашему мнению, в том, что механизмы обработки знаний рассматриваются в нем как чисто математическая задача, тогда как на самом деле её можно решить только на междисциплинарном уровне. Ключевые проблемы носят перцептивный характер и лежат в области когнитивной психологии, а совсем не в том, чтобы найти «еще более лучший» логический формализм. Математического инструментария накоплено

сегодня достаточно, но мало что дошло до широкого практического применения.

Нам представляется, что здесь мы как исследователи попадаем в некоторую ловушку «интеллектуальности», пытаясь смоделировать собственное мышление вместо того, чтобы оглянуться вокруг. Мы бьемся с автоматизацией доказательства теорем, разрабатываем сложные модели знаний и не обращаем внимания на то, что в основной массе решаемые людьми задачи намного проще. Например, ритейлер как бизнес-модель является исключительно сложной системой. Но эта модель устроена так, что каждая её компонента («рабочее место») вполне доступна для достаточно широкого круга людей и достаточно легко моделируется логически. Сложность модели ритейлера возникает тогда, когда эти локально-простые «рабочие места» используются в качестве кирпичиков для построения общего здания бизнес-модели. И здесь сложность может быть непомерной.

Наша гипотеза состоит в том, что огромное количество практически значимых моделей в мире являются локально-простыми, поэтому именно на них нужно обратить пристальное внимание. При этом особо внимательно нужно отнестись к обеспечению доступности логического формализма для «обычных» пользователей. Без этого нас будет ждать судьба семантического веба. Логика элитарна и труднодоступна, поэтому, в идеале, люди вообще не должны знать, что работают в рамках логической системы. Нужно найти привычную людям метафору, которая позволила бы им понимать, что происходит, без необходимости изучать логику. В качестве такой метафоры мы используем понятие документа.

Использование локально-простых моделей задает большие возможности для автоматизации управления знаниями, создания управленческих роботов и машинного обучения. Правда, популярные системы машинного обучения — нейронные сети — плохо подходят для работы в базах знаний, поскольку могут обучаться только через «натаскивание». Однако большие перспективы здесь имеют логико-вероятностные методы, например, разрабатываемые командой Е. Е. Витяева [10; 1]. Для этих методов семантические модели могут служить онтологиями.

3. Документные модели

В данной работе мы определяем понятие документной модели — версии исполняемой семантической модели, которая базируется на метафоре документа как базовом конструкте логических описаний. Документное моделирование реализует нашу концепцию управления системами знаний. Основные положения этой концепции изложены нами в работах [11; 2; 3].

Во-первых, документные модели являются исполняемыми. Это делает стадию программирования ненужной, поскольку сама модель может играть роль соответствующей информационной системы.

Во-вторых, модель использует понятие документа как метафору. Документная модель устроена как коллекция логических структур, которые могут быть интерпретированы как «обычные» документы, сохраняя при этом все возможности семантического моделирования и искусственного интеллекта. С другой стороны, для людей работа в такой модели похожа на привычную работу с документами.

Далее в работе дается математическое определение документной модели.

4. Система базовых типов

Пусть $\mathcal{B} = \langle B_1, \ldots, B_k;\ \Omega\rangle$, многосортная модель, определяющая базовые типы данных, где B_i – основные множества типов. На практике это могут быть строковые данные, целые и вещественные числа, изображения, видео и т.д. Сигнатура

$$\Omega = \langle \Omega_P, \Omega_F, \gamma\rangle$$

состоит из

- предикатных символов Ω_P
- функциональных символов Ω_F, а также
- функции γ, задающей местность предикатных и функциональных символов.

Будем считать, что все элементы всех сортов являются выделенными (могут быть представлены константами — нульместными функциональными символами из Ω_F). Константы будем обозначать через c, c_i. Элементы модели, соответствующие константам, обозначаются $\mathbf{c}, \mathbf{c}_i$.

Через

$$\mathbb{B} = \{\mathbf{b}^1, \ldots, \mathbf{b}^k, \mathbf{any}\}$$

обозначим множество *имен* базовых типов данных. Имя **any** обозначает тип всех элементов. Все предикатные и функциональные символы модели типизированы.

Типом предикатного символа $p, \gamma(p) = n$, назовем выражение

$$\langle \mathbf{b}_1, \ldots, \mathbf{b}_n\rangle,$$

где $\mathbf{b}_i \in \mathbb{B}$ означает, что i-й аргумент предиката, соответствующего p, должен принадлежать основному множеству B_i.

Аналогично, *типом функционального символа* $f, \gamma(f) = n$, назовем выражение

$$\langle \mathbf{b}_1, \ldots, \mathbf{b}_n, \mathbf{b}_{n+1}\rangle,$$

$\mathbf{b}_i \in \mathbb{B}$, где $\mathbf{b}_i$, $1 \leq i \leq n$, определяют тип аргументов, а $\mathbf{b}_{n+1}$ — тип результата соответствующей функции.

Понятия терма, атомарной формулы и типа терма определяются индуктивно стандартным образом.

5. Последовательности

Последовательностью назовем выражение

$$(e_1, \ldots, e_m),$$

где e_i — некоторые элементы. В дальнейшем в качестве элементов последовательностей будут выступать константы из Ω_F, а также ссылки на документы. На последовательностях выполняются следующие равенства

$$\begin{aligned} (e) &= e \\ (\ldots, (e_1, \ldots, e_k), \ldots) &= (\ldots, e_1, \ldots, e_k, \ldots) \end{aligned}$$

Первое равенство говорит о том, что одноэлементная последовательность неотличима от самого элемента. Второе правило гласит, что последовательности не имеют вложенности и являются плоскими (в отличие, например, от списков).

Допускается пустая последовательность, обозначаемая ( ).

Для управления количеством элементов в последовательности введем понятие кардинальности. Определим следующие кардинальности:

- ( ) — последовательность без элементов
- ? — последовательность, содержащая не более одного элемента
- ! — последовательность, состоящая ровно из одного элемента
- \+ — последовательность из одного или более элементов
- * — последовательность, состоящая из любого количества элементов

Через $\mathbb{C}$ обозначим множество всех кардинальностей.

6. Документы

Документ является основным понятием в документной модели. По своей роли документы аналогичны объектам в объектно-ориентированном подходе (ООП). Основные факты о документах:

1) Документ состоит из *полей*, для каждого из которых задан тип и кардинальность значений.

2) *Формы документов* являются шаблонами, которые описывают структуру документов определенного вида (некоторый аналог классов в ООП).

3) Документы могут ссылаться друг на друга через механизм *нумерации*.

4) Каждый документ обладает *статусом*, описывающим его текущее состояние. Переходы от статуса к статусу формируют жизненный цикл документа. Допустимые значения для статусов документов определены в их форме.

5) С помощью подмножества языка Libretto [4] (Librettino) можно определить автоматические правила перехода документа от статуса к статусу с совершением определенных операций на модели (*транзакции*, проводки документов).

6) Правила, которые задают *допустимые переходы* статусов документов, определяются в их форме.

Введем счетное множество новых констант

$$\mathbb{I} = \{id_1, id_2, \ldots\}$$

которое будем называть множеством имен (идентификаторов). Это множество разбивается на два непересекающихся счетных подмножества имен форм $\mathbb{I}_F$ и имен полей документов $\mathbb{I}_D$:

$$\mathbb{I}_F \cap \mathbb{I}_D = \emptyset$$

$$\mathbb{I}_F \cup \mathbb{I}_D = \mathbb{I}$$

В дальнейшем имена форм будут служить для описания типов документов. Поэтому множество всех имен типов данных определим как объединение имен базовых типов и имен форм:

$$\mathbb{B} \cup \mathbb{I}_F$$

Описанием поля документа назовем тройку

$$\mathbf{d} = \langle d, \mathbf{g}, \mathbf{c} \rangle,$$

где $d \in \mathbb{I}_D$ – имя поля, $\mathbf{g} \in \mathbb{B} \cup \mathbb{I}_F$ — его тип, и $\mathbf{c} \in \mathbb{C}$ – его кардинальность. Имена полей документов будем обозначать с помощью символа d, возможно с индексами. Описание поля документа, соответствующего имени d будем обозначать через $\mathbf{d}$.

Для удобства при определении полей будем использовать нотацию с использованием двоеточия. Например,

```
возраст:Int!
имена_детей:String*
```

вместо $\langle$возраст$, Int, !\rangle$ и $\langle$имена_детей$, String, *\rangle$, соответственно. Неформально, возраст это ровно одно целое число, а «имена детей» — произвольная последовательность строк.

Введем счетное множество новых констант, которые будем называть статусами документов:

$$\mathbb{S} = \{\mathbf{s}^1, \mathbf{s}^2, \ldots\}$$

Описанием транзакции назовем тройку

$$\mathbf{p} = \langle \mathbf{s}_{in}, \mathbf{s}_{out}, P(o) \rangle$$

Здесь

- $\mathbf{s}_{in}, \mathbf{s}_{out} \in \mathbb{S}$
- $\mathbf{s}_{in}$ называется начальным статусом транзакции,
- $\mathbf{s}_{out}$ называется конечным статусом, а
- $P(o)$ — код транзакции.

Код транзакции $P(o)$ представляет собой последовательность охраняемых операторов:

$$P(o) = \langle G_1(o) \to P_1(o); \ldots; G_k(o) \to P_k(o) \rangle$$

Он имеет в качестве единственного параметра документ o, проводка которого определяет транзакцию. Неформальная семантика последовательности охраняемых операторов следующая: последовательность равна крайнему левому $P_i(o)$, для которого истинен «гард» $G_i(o)$.

Неформально, данное определение задает транзакцию документа o, имеющего статус s_{in}, которая переводит его в статус s_{out} и выполняет набор инструкций, порожденных исполнением кода $P(o)$.

Пока мы не уточняем язык, на котором кодируются $P(o)$. Концептуально, это должен быть очень простой и слабый язык, обеспечивающий элементарность транзакций. Простота языка является важнейшим признаком *локальной простоты* моделей, которые мы будем строить.

Определим теперь понятие *формы документа*, которая описывает структуру однотипных документов. Формой документа назовем четверку

$$\mathbf{f} = \langle\, f, \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}, \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}, \{\mathbf{p}_1 \ldots \mathbf{p}_k\} \rangle$$

Здесь

- $f \in \mathbb{I}_F$ – имя формы;
- $\{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ — конечное множество описаний полей без повторения имен;
- $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ — конечное множество допустимых статусов;
- $\{\mathbf{p}_1 \ldots \mathbf{p}_k\}$ — конечное множество описаний транзакций.

Введем основное понятие данной работы — понятие *документа*. Для идентификации документов и доступа к ним используется *нумерация*.

Источником номеров для нас будет копия множества натуральных чисел $\mathbb{N}$. Чтобы отличать их от обычных целых чисел, будем записывать их с приставкой id, например, $id{:}n_1, id{:}5$. Номера документов будем называть *ссылками*. Документ, соответствующий ссылке $id{:}n$, обозначим через $\nu{:}n$. Если обозначить через $\mathbb{D}$ множество всех документов, то

$$\nu : \mathbb{N} \to \mathbb{D}$$

Теперь введем понятие *поля документа*, определенного как пара

$$\mathbb{d} = \langle d, w \rangle$$

где $d \in \mathbb{I}_D$ — имя поля, а w — последовательность допустимых значений. Допустимыми значениями полей являются элементы основных множеств $B_1, \ldots, B_k$ и ссылки из $\mathbb{N}$.

Будем говорить, что элемент e имеет тип $\mathbf{g}$ относительно нумерации ν, если выполняется одно из условий:

1) $\mathbf{g} = \mathbf{any}$

2) $\mathbf{g} = \mathbf{b}^i$ и $e \in B_i$

3) $\mathbf{g} = f$, $e = id{:}n$, и f является именем формы документа $\nu{:}n$.

Документом $\mathbb{o}$ назовем структуру

$$\mathbb{o} = \langle\, f, \{\mathbb{d}_1, \ldots, \mathbb{d}_n\}, \mathbf{s} \rangle$$

где $f \in \mathbb{I}_F$ — имя формы документа, $\{\mathbb{d}_1, \ldots, \mathbb{d}_n\}$ — множество полей, а $\mathbf{s} \in \mathbb{S}$. Будем говорить, что документ $\mathbb{o}$ имеет статус $\mathbf{s}$, и обозначать это через $\mathbb{o}[\mathbf{s}]$.

Пусть σ — некоторая синтаксическая структура (например, форма или документ). Определим операции $id_F(\sigma)$ и $id_D(\sigma)$, которые равны множеству всех имен форм и всех имен полей, встречающихся в σ, соответственно. Также определим

$$\begin{aligned} id_F(\{\sigma_1, \ldots, \sigma_m\}) &= id_F(\sigma_1) \cup \ldots \cup id_F(\sigma_m) \\ id_D(\{\sigma_1, \ldots, \sigma_m\}) &= id_D(\sigma_1) \cup \ldots \cup id_D(\sigma_m) \end{aligned}$$

Сигнатурой документной модели назовем конечное множество форм документов

$$\mathbb{M} = \{\mathbf{f}_1, \ldots, \mathbf{f}_l\}$$

замкнутое относительно имен: $id_F(\mathbb{M}) \subseteq \{f_1, \ldots, f_l\}$, где f_i — имя формы $\mathbf{f}_i$.

Документной моделью назовем конечное множество документов

$$\mathcal{M} = \langle \{\mathbb{o}_1, \ldots \mathbb{o}_m\}, \nu \rangle$$

вместе с функцией ν, определяющей нумерацию на документах.

Будем говорить, что $\mathcal{M}$ является *моделью сигнатуры* $\mathbb{M}$, если для каждого документа $\mathbb{o} \in \mathcal{M}$, имеющего форму с именем f, выполняется:

1) $\mathbf{f} \in \mathbb{M}$, т. е. форма с именем f определена в сигнатуре $\mathbb{M}$;

2) Для любого поля $\mathbb{d} = \langle d, w \rangle$ документа $\mathbb{o}$ в форме $\mathbf{f}$ имеется описание $\mathbf{d} = \langle d, \mathbf{g}, \mathbf{c} \rangle$, причем размер последовательности w соответствует кардинальности $\mathbf{c}$, и каждый элемент из w имеет тип $\mathbf{g}$;

3) Статус $\mathbf{s}$ документа $\mathbb{o}$ является допустимым статусом формы $\mathbf{f}$.

Предложение 1. *Следующие соотношения верны:*
– $id_F(\mathcal{M}) \subseteq id_F(\mathbb{M})$
– $id_D(\mathcal{M}) \subseteq id_D(\mathbb{M})$.

7. Транзакции

Через механизм транзакций (проводок документов) модель $\mathcal{M}$ развивается и модифицируется во времени. Транзакции выполняются последовательно. Каждая новая транзакция определяет следующий момент времени жизненного цикла модели. Для формализации данного механизма введем упорядоченное счетное множество

$$\mathbb{T} = \{\mathbf{t_0}, \mathbf{t_1}, \mathbf{t_2}, \ldots\}$$

которое будем называть множеством моментов. $\mathbf{t_0}$ будем называть первоначальным моментом модели (моментом её «рождения»). Состояние модели на момент времени $\mathbf{t}_i$ будем обозначать через $\mathcal{M}^{\mathbf{t}_i}$. Общая схема применения транзакции с определением $\langle \mathbf{s}, \mathbf{s}', P(o) \rangle$ при проводке документа $\mathbb{o}$ выглядит следующим образом:

Правило 1. Проводка документа

$$\frac{\mathcal{M}^{\mathbf{t}_i}[\mathbb{o}[\mathbf{s}_{in}]] \qquad \langle \mathbf{s}_{in}, \mathbf{s}_{out}, P(\mathbb{o}) \rangle}{\mathcal{M}^{\mathbf{t}_{i+1}}[\mathbb{o}[\mathbf{s}_{out}]]}$$

Состояние модели $\mathcal{M}^{\mathbf{t}_{i+1}}$ получается из состояния $\mathcal{M}^{\mathbf{t}_i}$ выполнением инструкций, сгенерированных кодом $P(\mathbb{o})$.

Правило 2 формализует возможность внешнего воздействия на модель. Документная модель, как правило, не является изолированной. Она погружена в некоторый контекст, например, реальный мир. Из контекста в модель может приходить разнообразная информация – в виде новых документов или измененных значений полей документов.

На практике это может быть, например, ввод пользователем данных через веб-интерфейс, публикация в модели результатов работы системы машинного обучения и т.д.

Источники информации, подобным образом влияющие на модель, называются *оракулами*. Взаимодействие с оракулом также осуществляется через механизм транзакций. Каждый акт взаимодействия — это отдельная транзакция, выполняющая код, предоставленный оракулом. Правило взаимодействия с оракулом выглядит так:

Правило 2. Воздействие оракула

$$\frac{\mathcal{M}^{\mathbf{t}_i} \qquad P_{oracle}}{\mathcal{M}^{\mathbf{t}_{i+1}}}$$

Здесь P_{oracle} — код, предоставленный оракулом для выполнения. Состояние модели $\mathcal{M}^{\mathbf{t}_{i+1}}$ получается из состояния $\mathcal{M}^{\mathbf{t}_i}$ применением инструкций, порождаемых P_{oracle}.

Правила 1 и 2 выполняются следующим образом:

1) Код C выполняется в контексте модели $\mathcal{M}^{\mathbf{t}_i}$

2) Код C генерирует конечный набор инструкций $ins_1, \ldots, ins_k$.

3) Инструкции последовательно применяются к модели $\mathcal{M}^{\mathbf{t}_i}$, модифицируя её до состояния $\mathcal{M}^{\mathbf{t}_{i+1}}$.

4) Если все инструкции применены успешно, то правило применено, и модель переходит в состояние $\mathcal{M}^{\mathbf{t}_{i+1}}$.

5) Если выполненение некоторой инструкции закончилось неудачей, то правило неприменимо, и модель остаётся в состоянии $\mathcal{M}^{\mathbf{t}_i}$.

Таким образом, код $P(o)$ не оказывает непосредственного влияния на модель. Он генерирует конечную последовательность инструкций. Затем в рамках транзакции эти инструкции последовательно применяются к модели, переводя её в новое состояние. Транзакция, очевидно, должна обладать свойством атомарности — будут выполнены либо все инструкции, либо ни одной (например, если в процессе выполнения инструкции произошла ошибка, необходимо «откатить» вычисления к первоначальному состоянию).

Разделение выполнения транзакции на два этапа – порождения инструкций и применения инструкций – позволяет нам задать альтернативное монотонно расширяющееся определение модели. Представим транзакцию как тройку

$$\mathbb{p}_i = \langle \mathbf{t}_i, \langle \mathbf{s}_{in}, \mathbf{s}_{out}, P(o) \rangle, [ins_1, \ldots, ins_k] \rangle$$

для первого правила, и

$$\mathbb{p}_i = \langle \mathbf{t}_i, P_{oracle}, [ins_1, \ldots, ins_k] \rangle$$

для второго. Здесь $\mathbf{t}_i$ — момент времени, генерируемый транзакцией, $\mathbb{o}$ — проводимый документ для первого правила, а $ins_1, \ldots, ins_k$ — инструкции, исполненные в рамках транзакции.

Теперь состояние модели в момент $\mathbf{t}_n$ может быть неявно представлено как пара

$$\langle \mathcal{M}^{\mathbf{t}_0}, [\mathbb{p}_1, \ldots, \mathbb{p}_n] \rangle,$$

где $\mathcal{M}^{\mathbf{t}_0}$ – первоначальное состояние модели (как, правило, пустая модель).

Предложение 2. *Явное состояние модели $\mathcal{M}^{\mathbf{t}_n}$ в момент $\mathbf{t}_n$ можно получить последовательным применением всех инструкций из транзакций $\mathbb{p}_1, \ldots, \mathbb{p}_n$.*

8. Инструкции

Для успешной работы с моделью достаточно иметь очень простой набор, состоящий из трех типов инструкций:

1) `newdoc(formname)` — создание пустого документа соответствующей формы.

2) `set(doc, field, value)` — задание значения `value` полю `field` документа `doc`.

3) `state(doc, s)` — задание документу `doc` статуса `s`.

На практике полезно иметь более широкий спектр инструкций, но теоретически при достаточной выразительности языка порождения инструкций, этих трех достаточно.

Качества языка порождения инструкций, на котором разрабатывается код транзакций $P(o)$, играют ключевую роль. Мы можем задействовать в этом месте языки с самым разным уровнем выразительности, вплоть до полных по Тьюрингу языков. Однако мы намеренно выбираем достаточно простой язык, обеспечивающий разрешимость основных проблем и допускающий эффективный автоматический анализ кода. Это является одним из основных признаков локальной простоты модели.

Еще одним важным качеством языка является его декларативность. Данный язык должен иметь четкую декларативную семантику, обеспечивающую возможность автоматического анализа кода транзакций, например, в процессе использования искусственного интеллекта для

контроля за корректностью умных контрактов. Отметим, что разделение этапа порождения инструкций и их исполнения позволяет нам использовать декларативный язык при формировании инструкций, которые в дальнейшем изменят модель во вполне процедурном стиле.

В bSystem – нашей системе управления документными моделями, основанной на формализме, представленном в данной работе – в качестве языка генерирования инструкций выбрано простое декларативное подмножество языка Libretto [4]. Ниже мы разберем пример его использования.

9. Бизнес-процессы

Модель бизнес-процесса – тройка

$$\langle f, s_{beg}, s_{fin} \rangle$$

где $s_{beg}, s_{fin} \in \mathbb{S}$ — допустимые статусы формы с именем f, называемые, соответственно, начальным и конечным статусами бизнес-процесса.

Бизнес-процесс, реализующий модель $\langle f, s_{beg}, s_{fin} \rangle$, – это последовательность транзакций документа $\mathbb{o}$ формы **f** (называемого основным документом бизнес-процесса)

$$\mathbb{o}[s_{beg}] \to \mathbb{o}[s_1] \to \ldots \to \mathbb{o}[s_n] \to \mathbb{o}[s_{fin}]$$

стартующая когда $\mathbb{o}$ имеет начальный статус s_{beg} и переводящая $\mathbb{o}$ в конечный статус, причем для любого $i, 1 \leq i \leq n$, выполняется $s_i \neq s_{fin}$.

10. Пример: управление книжной библиотекой

В качестве примера построим простую модель управления книжной библиотекой, которая описывает два вида объектов — книги и читателей, а также операции выдачи и возвращения книг. Сначала опишем сигнатуру модели. Математическая нотация не очень удобна для читабельного представления описаний. Так, например, будет выглядеть определение формы «Книга»:

$$\langle \text{Книга}, \{\langle , \mathit{String}, ! \rangle, \langle , \mathit{String}, ! \rangle, \langle , , * \rangle\}\{,\}, \emptyset \rangle$$

Поэтому будем использовать более структурированный синтаксис:

```
Form Книга {
  автор: String!
  название: String!
  выдачи: Выдача*
  STATES: вколлекции, наруках
}
```

Так определяются библиографические карточки, содержащие информацию о книге, а также список выдач книги (поле `выдачи`). В зависимости от того, где находится книга, библиокарточка может иметь один из двух статусов. Определим теперь формы документов для читателя и выдачи.

```
Form Читатель {
  фамилия: String!
  имя: String!
  STATES: нормальный, задолжник
}

Form Выдача {
  книга: Книга!
  читатель: Читатель!
  STATES: подготовка, выдана, возвращена
}
```

Процесс выдачи книги состоит из трех этапов (подготовка к выдаче, книга выдана, книга возвращена), которые отражены в статусах документа. Переход от статуса к статусу реализуется через транзакции документа `Выдача`, определенные следующим образом

```
Выдача: подготовка -> выдана {

  case (книга.state == вколлекции) => {
    set(книга, выдана, (книга.выдана, this))
    state(книга, наруках)
  }

  case _ => error(''Повторная выдача!'')
}

Выдача: выдана -> возвращена {

  case (книга.state == наруках) =>
    state(книга, вколлекции)

  case _ => error(''Уже в коллекции'')
}
```

Транзакции исполняются в контексте конкретного документа выдачи, в котором определены поля книги и читателя. При первом переходе (выдаем книгу) генерируется инструкция присоединения документа

выдачи (this) к полю выдачи документа Книга, а также инструкция изменения статуса документа Книга. При втором переходе (возвращаем книгу) генерируется инструкция изменения статуса документа книги. Кроме того, автоматически генерируются инструкции изменения статуса самого документа выдачи.

Эксплуатацию начнем с пустой модели. Сначала в рамках первой транзакции добавим в модель книгу и читателя (это правило воздействия оракула, поскольку информация приходит извне):

```
t-1 { // оракул
  newdoc(Книга, вколлекции) // id:1
  set(doc:1, автор, ''Пушкин'')
  set(doc:1, название, ''Евгений Онегин'')

  newdoc(Читатель, нормальный) // id:2
  set(doc:2, фамилия, ''Иванов'')
  set(doc:2, имя, ''Иван'')
  state(doc:2, нормальный)
}
```

Через doc:i обозначается документ — значение $\nu(i)$. В рамках второй транзакции формируем документ выдачи Пушкина Иванову (это правило оракула – работа библиотекаря):

```
t-2 { // оракул
  newdoc(Выдача, оформление)  // id:3
  set(doc:3, книга, doc:1)
  set(doc:3, читатель, doc:2)
  state(doc:3, оформление)
}
```

На практике эти две транзакции могут проводиться библиотекарем, например, через веб-интерфейс. Факт выдачи книги фиксируется правилом проводки документа для doc:3:

```
t-3 { // проводка doc:3 оформление -> выдана
  state(doc:3, выдана)
  set(doc:1, выдана, (( ), doc:3))
  state(doc:1, наруках)
}
```

Этот набор инструкций порожден кодом транзакции. Поскольку doc:3 имеет статус оформление, то сработает первое определение транзакции, которое порождает инструкции, прицепляющие документ выдачи к книге и меняющие статусы документов выдачи и книги.

Когда книга возвращена документ doc:3 проводится снова:

```
t-4 { // проводка doc:3 выдана -> возвращена
  state(doc:3, возвращена)
  state(doc:1, вколлекции)
}
```

Поскольку в этом случае статус `doc:3` равен `выдана`, то на сей раз срабатывает второе правило для транзакции. Документ выдачи переходит в статус `возвращена`, а сама книжка – в статус `вколлекции`. Бизнес-процесс выдачи книжки Пушкина завершен.

11. Умные контракты

Модель вместе с определениями транзакций содержит все инструменты для формирования *умных контрактов* (smart contracts). Умные контракты реализуют автоматическое управление взаимодействием контрагентов с обеспечением децентрализованных средств доверия между ними на основе криптографических технологий.

По сути, модель библиотеки выше является примером умного контракта между библиотекой и читателем. Чтобы сделать его полноценным, нужно обеспечить механизм доверия. Это можно сделать, сопрягая документную модель с блокчейн-платформой, на которой будут сохраняться все транзакции, связанные с проводками документов `Выдача`. Библиотекарь и читатель могут скреплять проводки своими криптоподписями.

В рамках представленного нами формализма можно дать следующее определение умного контракта:

Определение 1. *Умный контракт – бизнес-процесс, инструкции которого сохраняются в блокчейне.*

Действительно, документные модели предоставляют инструменты, необходимые для формулирования и интеллектуального управления исполнением умных контрактов через механизмы транзакций. Необходимо только обеспечить фиксацию процесса исполнения, не позволяющую изменять прошлое.

12. Заключение

Сейчас основные наши усилия направлены на апробирование документных моделей на практических задачах реальной сложности. Мы научились эффективно работать с моделями, содержащими десятки миллионов документов и более. В частности, мы применяем документные модели в сервисах управления бизнес-процессами — там, где царят

системы SAP R3, Oracle ERP, Microsoft Dynamic Ax, 1C, Qlik, Salesforce и др. В сотрудничестве с нашими бизнес-партнерами мы внедряем такие проекты как система бюджетирования и управления движением товара в ритейлере, CRM-модель, коммерческую отчетность и управление персоналом в торговле, управление производством продуктов, ERP-модель и др.

Список литературы

1. Витяев Е. Е. Семантический вероятностный вывод предсказаний / Е. Е. Витяев // Изв. Иркут. гос. ун-та. Сер. Математика. — 2017. – Т. 21.
2. Малых А. А. Логические архитектуры и объектно-ориентированный подход / А. А. Малых, А. В. Манцивода, В. С. Ульянов // Вестн. НГУ. Сер. Математика, механика, информатика. – 2009. – Т.9, вып. 3. – С. 64-85.
3. Малых А. А. Объектные теории над списочными надстройками / А. А. Малых, А. В. Манцивода // Изв. Иркут. гос. ун-та. Сер. Математика. – 2012. – № 4. - С. 27–44.
4. Малых А. А. Система Libretto: разработка веб-ресурсов в единой модели данных и знаний / А. А. Малых, А. В. Манцивода // 6-я Всерос. конф. по проблемам управления (МКПУ-2013). Геленджик, 30 сент. – 5 окт. 2013 г.– С. 73-75.
5. Berners-Lee T. The Semantic Web / Lee T. Berners, J. Hendler, O. Lassila// Scientific American. – May, 2001. https://doi.org/10.1038/scientificamerican0501-34
6. Goncharov S. S. Semantic foundations of programming / S. S. Goncharov, Yu.L. Ershov, D.I. Sviridenko // Lecture Notes in Computer Science, 1987. – Vol. 278. - P. 116-122. https://doi.org/10.1007/3-540-18740-5_28
7. Goncharov S. S. Semantic programming / S. S. Goncharov, Yu. L. Ershov, D. I. Sviridenko // Information processing, Proc. IFIP 10-th World Comput. Congress. – Dublin, v. 10, 1986. – P. 1093–1100.
8. Horrocks I. From SHIQ and RDF to OWL: The making of a Web Ontology Language / I. Horrocks, P. Patel-Schneider, F. Van Harmelen // Journal of Web Semantics.- Vol. 1, Issue 1.- P. 7–26. https://doi.org/10.1016/j.websem.2003.07.001
9. Horrocks I. Practical reasoning for expressive description logics / I. Horrocks, U. Sattler, U. Tobies // Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), number 1705 in Lecture Notes in Artificial Intelligence / eds.: H. Ganzinger, D. McAllester, A. Voronkov. – Springer-Verlag, 1999. – P. 161-180. https://doi.org/10.1007/3-540-48242-3_11
10. Kovalerchuk B. Data Mining in finance: Advances in Relational and Hybrid Methods / B. Kovalerchuk, E. Vityaev. – 2000. – Kluwer Academic Publishers, 456 p.
11. Malykh A. Query Language for Logic Architectures / A. Malykh, A. Mantsivoda // Lecture Notes in Computer Science 5947. – Springer-Verlag, Berlin Heidelberg. – 2010. – P. 294–305. https://doi.org/10.1007/978-3-642-11486-1_25
12. Robinson J. A. A Machine Oriented Logic Based on the Resolution Principle // J ACM. – 1965. – Vol. 12. – P. 23-41. https://doi.org/10.1145/321250.321253
13. Riazanov A. The design and implementation of VAMPIRE / Riazanov A., Voronkov A. // Journal AI Communications. – 2002. – Vol. 15, Issue 2,3. – P. 91–110.
14. Semantic Web activity [Electronic resource]. – URL: http://www.w3.org/2001/sw/.

Манцивода Андрей Валерьевич, доктор физико-математических наук, профессор, Иркутский государственный университет, 664003,

г. Иркутск, ул. К. Маркса, 1, Институт математики им. С.Л. Соболева, 630090, г. Новосибирск, пр. ак. Коптюга, 4 (e-mail: andreibaikal.ru)

Малых Антон Александрович, кандидат физико-математических наук, Иркутский государственный университет, 664003, г. Иркутск, ул. К. Маркса, 1 (e-mail: malykhbaikal.ru)

A.A. Malykh, A.V. Mantsivoda
Document Models

Abstract. In this paper, the concept of locally simple models is considered. Locally simple models are arbitrarily complex models built from relatively simple components. A lot of practically important domains of discourse can be described as locally simple models, for example, business models of enterprises and companies. Up to now, research in human reasoning automation has been mainly concentrated around the most intellectually intensive activities, such as automated theorem proving. On the other hand, the retailer business model is formed from "jobs", and each "job" can be modelled and automated more or less easily. At the same time, the whole retailer model as an integrated system is extremely complex. In this paper, we offer a variant of the mathematical definition of a locally simple model. This definition is intended for modelling a wide range of domains. Therefore, we also must take into account the perceptual and psychological issues. Logic is elitist, and if we want to attract to our models as many people as possible, we need to hide this elitism behind some metaphor, to which 'ordinary' people are accustomed. As such a metaphor, we use the concept of a document, so our locally simple models are called document models. Document models are built in the paradigm of semantic programming. This allows us to achieve another important goal - to make the documentary models executable. Executable models are models that can act as practical information systems in the described domain of discourse. Thus, if our model is executable, then programming becomes redundant. The direct use of a model, instead of its programming coding, brings important advantages, for example, a drastic cost reduction for development and maintenance. Moreover, since the model is well and sound, and not dissolved within programming modules, we can directly apply AI tools, in particular, machine learning. This significantly expands the possibilities for automation and robotization of management and control activities.

Keywords: locally simple model, document model, semantic programming.

References

1. Vityev E. Semantic Probablistic Inference of Predictions. *Izv. Irkutsk. Gos. Univ. Ser. Mat.*, 2017, vol. 21. (in Russian)
2. Malykh A., Mantsivoda A., V.S.Ulyanov. Logical Architectures and Object Oriented Approach. *Vestnik NGU. Series: Mathematics, Mechanics, Informatics.*, 2009, vol.9, issue 3, pp. 64-85. (In Russian)
3. Malykh A., Mantsivoda A. Object Theories over List Superstructures. *Izv. Irkutsk. Gos. Univ. Ser. Mat.*, 2012, no 4, pp. 27-44. (in Russian)
4. Malykh A., Mantsivoda A. Sistema Libretto: razrabotka web-resursov v edinoi modeli dannykh i znanii [Libretto System: Web Resources Development Based On a Holistic Data and Knowledge Model]. In: *Proceedings of The 6th All-Russian Conference on Control Problems*, Gelendzhik, September, pp.73-75.

5. Berners-Lee T., Hendler J., Lassila O. The Semantic Web. *Scientific American*, May 2001. https://doi.org/10.1038/scientificamerican0501-34
6. Goncharov S.S., Ershov Yu.L., Sviridenko, D.I. Semantic foundations of programming. *Lecture Notes in Computer Science*, 1987, vol. 278, pp. 116-122. https://doi.org/10.1007/3-540-18740-5_28
7. Goncharov S.S., Ershov Yu.L., Sviridenko D.I. Semantic programming. In: *Information processing, Proc. IFIP 10th World Comput. Congress*, Dublin, 1986, vol.10, pp.1093-1100.
8. Horrocks I., Patel-Schneider P., Van Harmelen F. From SHIQ and RDF to OWL: The making of a Web Ontology Language. *Journal of Web Semantics*, v.1, no 1, pp.7–26. https://doi.org/10.1016/j.websem.2003.07.001
9. Horrocks I., Sattler U., Tobies U. Practical reasoning for expressive description logics. In: *H. Ganzinger, D. McAllester, and A. Voronkov, editors, Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, no 1705 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1999, pp. 161-180. https://doi.org/10.1007/3-540-48242-3_11
10. Kovalerchuk B., Vityaev E. Data Mining in finance: Advances in Relational and Hybrid Methods. *Kluwer Academic Publishers*, 2001. 456 p.
11. Malykh A., Mantsivoda A.A. Query Language for Logic Architectures. *Lecture Notes in Computer Science*, no 5947, Springer-Verlag, Berlin Heidelberg, 2010, pp.294–305. https://doi.org/10.1007/978-3-642-11486-1_25
12. Robinson J.A. A Machine Oriented Logic Based on the Resolution Principle. *J ACM*, 1965, no 12, pp.23-41. https://doi.org/10.1145/321250.321253
13. Riazanov A., Voronkov A. The Design and Implementation of VAMPIRE. *Journal AI Communications*, 2002, vol. 15, issue 2,3, pp.91-110.
14. Semantic Web activity. http://www.w3.org/2001/sw/.

Mantsivoda Andrei Valerievich, Doctor of Sciences (Physics and Mathematics), Professor, Irkutsk State University, 1, K. Marx, Irkutsk, 664003,Sobolev Institute of Mathematics, 4 Acad. Koptyug avenue, Novosibirsk, 630090 (e-mail: andrei@baikal.ru)

Malykh Anton Alexandrovich, Candidate of Sciences (Physics and Mathematics), Irkutsk State University, 1, K. Marx, Irkutsk, 664003, (e-mail: malykh@baikal.ru)